Semantic Word Processing for Content Authors

Marcelo Tallis
Teknowledge Corporation.
4640 Admiralty Way
Marina del Rey, CA, USA
email mtallis@teknowledge.com

ABSTRACT

Document authors cannot routinely afford the overhead imposed by current semantic annotation tools. Some characteristics of their task can be exploited to provide them with a tool that will reduce the effort required to create both the document content and their accompanying semantic annotations.

SemanticWord is such a semantic annotation tool. SemanticWord is an environment based in MS Word that integrates content and markup authoring, providing customizable tools that allow *simultaneous* generation of content and semantic annotations, an annotation scheme that allows annotations to be reused when content is reused, a customizable library of templates containing partially annotated text, and an automatic information extraction system with the tools for refining and augmenting its output.

Keywords

SemanticWord, Semantic Annotations, Semantic Web, Markup Authoring Tools, COTS integration.

INTRODUCTION

The vast amount of information contained in the web is beyond any individual's grasp. Unfortunately, its content is primarily tailored to human consumption and not suitable for automatic semantic interpretation. The semantic web addresses this problem by allowing content to be annotated with machine understandable semantic descriptions.

Although current annotation tools take care of the annotations syntax and the proper reference and use of ontology terms ([4][5][6]), authoring semantic annotations remains a tedious and expensive process.

While this cost may be affordable to people who author web documents sporadically (e.g., a teacher authoring her homepage) it would be prohibitive to those who author and update documents routinely (e.g., an intelligence analyst writing intelligence reports).

Automatic Information Extraction systems have been suggested as an alternative method for generating semantic annotations. Unfortunately, this technology is only able to extract sufficient information to fill in a flat template and cannot capture the relationship graph that connects the instances ([5][7]).

Clearly, current markup authoring tools are inadequate for the task of routinely authoring content. Fortunately, some characteristics of this task, as it applies to some authors, can be exploited to reduce the cost of producing these annotations. Some of these characteristics are:

- The documents to be authored are primarily confined to a few topics. In this case it is worthwhile to spend some effort in setting up an environment tailored to these topics. The savings from producing multiple documents will more than recoup the tailoring cost.
- There is a high degree of content reuse. For example, different documents include common actors and places, share the same context, or update on previous accounts. This characteristic can be exploited to reuse the annotations along with the content.

SemanticWord is a semantic annotation tool designed with this kind of task in mind. Some of the features included in SemanticWord are:

- An environment that integrates content and markup authoring. This environment is based in MS Word, a product that is already familiar to many authors.
- Customizable tools for simultaneous generation of content and semantic annotations.
- An annotation scheme that allows for annotations to be reused when content is reused.
- A customizable library of templates containing partially annotated text. Authors can include templates in their documents to speed up both content and annotation production.
- An automatic information extraction system and the tools for refining and augmenting its output.

SEMANTIC WORD

SemanticWord offers an environment for authoring annotated text documents based in MS Word. Its aim is to reduce the burden involved in authoring semantic annota-

tions. Authors are given a familiar and uniform environment where the creation of content and semantic descriptions can be freely interleaved. In many case both of them can be achieved in a single operation.

Overview

SemanticWord extends MS Word in several dimensions (see Figure 1). First, MS Word GUI is augmented with toolbars that support the creation of semantic descriptions (or annotations) that are attached to text regions. The GUI is also extended to show these annotations embedded within the text and to support their direct manipulation through mouse gestures. Second, SemanticWord extends Word's reach by opening a channel to the Semantic Web. Content from the Semantic Web (both ontology definitions and factual descriptions) is brought into SemanticWord to compose annotations that are later dumped back into the Semantic Web. Third, SemanticWord extends Word services by integrating AeroDAML, an automated information extraction system. AeroDAML analyzes and annotates the text of the document as it is being typed, appearing to the author as a service analogous to Word's spelling and grammar checking. Finally, SemanticWord supports the rapid composition of annotated text through template instantiation.

The above extensions were implemented using standard Microsoft extensibility technology. Annotations are rendered with ActiveX controls that can be placed in a document, implement their own behavior, control their GUI, and save their internal state. Automatic text analysis is driven by SmartTags technology that supports background parsing and tagging of the document text as it is being typed. The rest is supported by an Office COM Add-in that responds to MS Office/MS Word built-in events (e.g., DocumentOpen) and extend Word's menus and toolbars. The document content is manipulated through Word's COM API.

Semantic Annotations

SemanticWord annotations are based in the DAML+OIL language [3]. DAML+OIL is a knowledge representation language developed for the Semantic Web that supports the definition of machine-readable ontologies and the linking of terms in documents to ontologies.

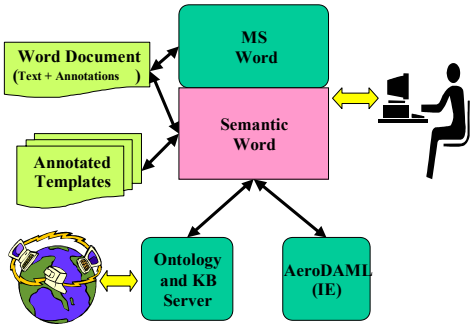


Figure 1. SemanticWord Architecture

SemanticWord annotations are attached to regions of text, not to the document as a whole. There are two types of annotations: *instances references* and *triple bags*. An instances reference associates a text region with a "referencable" instance of a class. Triple bags describe the content of a text region with a collection of triples that follow DAML+OIL's *subject-predicate-object* model. The subject is an instance, the predicate is a property defined in an ontology, and the object can either be an instance or a value. SemanticWord Annotations are retained across text copy/cut and paste operations.

Figure 2 illustrates a fragment of an annotated document. An instance reference is rendered by enclosing the annotated text between square brackets and with an icon adjacent to the closing bracket. A triple bag is rendered by enclosing the annotated text between square brackets and displaying a checkbox and a triples table adjacent to the closing bracket. The checkbox allows the user to display or hide the table. To facilitate the handling of heavily annotated documents, the text associated to an individual annotation can be highlighted and all annotation marks can be made invisible.

An Instance reference icon can be dragged and dropped over a cell corresponding to the subject or object of a triple. Cells filled using this method do not store a direct reference to the dropped instance but rather establish a link with the dragged instance reference. Updating the linked instance reference to refer to a different instance will alter the triple too. This level of indirection improves maintainability.

Triple cells can also be filled by picking instances and properties from special purpose browsers called *choosers* (See Figure 3). Choosers can use the values already stored in a triple to constrain the lists of choices offered to the user. For example, if the subject and object of a row are already filled in then the corresponding property chooser will only show the properties whose domain and range are consistent with those entries. Because SemanticWord does not enforce consistency, these constraints can be relaxed. The choosers also provide other filters for constraining the choices shown. For example, the instance chooser includes filters for listing only the instances that have already been referenced in the document. The instance choosers can selectively list instances corresponding to preexisting semantic web markup (provided by the Ontology and KB Server) or new instances defined in the current document. They also allow users to *create* new locally defined instances or provisional instances (described below), a function that a user would invoke if the listed choices do not include the desired instance. SemanticWord does not impose any order for filling in table cells, and can persist the state of tables containing rows with one or more empty cells.

Locally defined instances are instances that cannot be referenced from outside the document. Provisional instances are an artifact to postpone the identification of an instance that is being used to describe relationships. Ultimately, provisional instances must be replaced by references to external or locally defined instances. SemanticWord keeps track of the provisional instances and assists users in replacing them.

One obstacle that we noticed in other systems when composing a triple is that the role of the instances in the triple cannot be established before examining the definition of the predicate property. For example, determining who is the subject and who is the object in the relationship between an employee and her employer depends on how the property that relates both of them is (arbitrarily) defined. Assigning an instance to the subject or the object of a triple prematurely might preclude the possibility of establishing the relationship. To avoid this problem in SemanticWord, the property chooser can optionally list *reversed properties*. Reversed properties are ordinary properties that assume the subject and the object of a triple are switched. Reversed properties is only an artifact to add another degree of liberty in the order in which the triple arguments are filled -- the generated DAML markup switches the subject and object of a triple when a reversed property was selected.

Taming Annotation Authoring

SemanticWord was conceived with the goal of minimizing the burden involved in authoring semantic annotations. This burden is reduced through several techniques.

Non Intrusive Annotation Environment

SemanticWord provides an environment for authoring semantic annotations that is tightly integrated to MS Word. Word is the most massively adopted product for authoring text documents. SemanticWord includes a set of tools that economize the production of semantic descriptions and exploit opportunities for the simultaneous generation of text and annotations. Two examples of these tools are *personal class toolbars* and the *cascading class menus,* both illustrated in Figure 4.

Personal Class Toolbars: Personal Class Toolbars constitutes a convenient tool for generating both content and annotations together with just one mouse click. Users can create any number of Personal Class Toolbars, each one of them tied to a single class. Each personalized class toolbar contains an *instance selection* combo box and buttons to create instance references corresponding to the selected instance or a new one. If at the time the user creates an instance reference the document contains a selected region of text, then the instance reference will be attached to that region. If no text is currently selected, then both the "label" of the instance reference will be inserted in the document at the current text insertion point, and the new reference will be associated with the inserted text.

Personal class toolbars save effort when a small percentage of classes or instances account for a substantially larger percentage of the instance references that an author will need.

Classes Cascading Menu: A cascading class menu includes an entry for every named class in the ontology attached to the document. This menu gives users access to most of the operations related to ontology classes, including defining new instances, creating personal class toolbars, and opening instance choosers. When a user executes

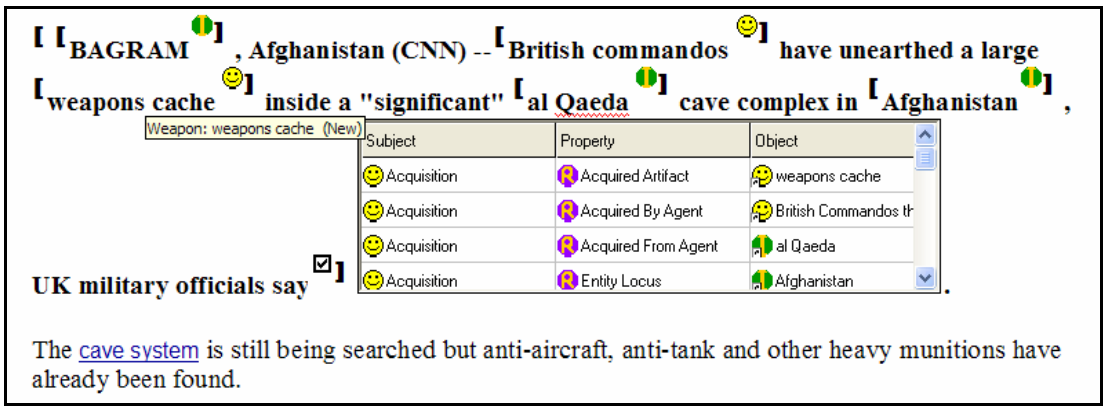


Figure 2. Fragment of an annotated document.

The circular icon containing an I Bar (like that adjacent to "BAGRAM") references an external instance from the semantic web. A smiley face icon (like that adjacent to "weapons cache") references a locally defined instance. The boxed legend below the "weapons cache" instance reference is its tool tip. If the instance icon in the subject or object column of a table is overlapped by a small arrow in its lower left corner (like the one in the object column of the first row) then the cell is linked to an instance reference annotation. Modifying or deleting the linked instance reference will affect the triple too. If the instance icon is not overlapped by a small arrow (like the one the subject column of the first row) then the cell contains a direct reference to an instance.

any of these functions from this menu, the menu entry corresponding to the selected class is duplicated and placed at the top of the menu so the user can access it easily the next time that she needs it. The cascading hierarchy is determined by the subclass hierarchy of the ontology. Classes with multiple superclasses appear in the cascade under each superclass.

Direct Manipulation of Annotations: Direct manipulation of annotations is another method of simplifying the production of semantic annotations. In SemanticWord users can compose semantic annotations by manipulating other annotations that are placed within the document. For example, the subject and object of a triple can be filled by dragging instance references annotations over the triple. For some users this method is faster and more natural than searching for those same instances in instance browsers.

Flexible commitment order

Authors should not be forced to follow a strict order in carrying out the many steps involved in authoring semantic descriptions. Many of the features that support this principle have been introduced before. These features are summarized in this section.

- Elements of a triple can be entered in any order. Even the determination of which instance is the subject and which is the object can be postponed by means of the **reversed properties**. New instances can be created from the instance choosers avoiding a disruption of the triple's composition process. Unlike other annotation tools, triples are laid out in a tabular structure rather than in a tree or other structures that impose a topological dependency among its nodes.
- Consistency is not enforced. A user is free to compose a triple that violates ontology constraints. The user can make the changes that would fix this conflict at a time convenient to her. Consistency is taken into account when filtering suggested choices for composing a triple, but the user can deactivate these filters with a single button click.
- Instance identification can be postponed but the instance can still be used to describe relationships. This is achieved through the use of **provisionary instances**, which can be used wherever definitive instance can but remind the user of the unconcluded task. SemanticWord will assist users in assigning identity to these instances.

Annotation Reuse

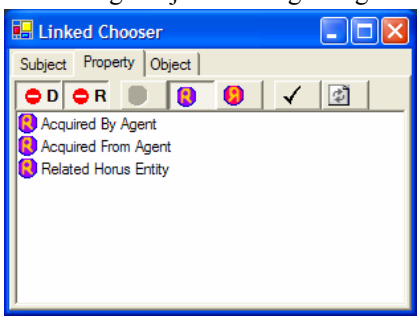
Annotations are attached to text regions and are going to be reused when those regions are reused. In particular, annotations are carried over along text cut/copy and paste operations and when fragments of a document are reused elsewhere in the same document in other documents based on the same ontology.

Automatic Information Extraction

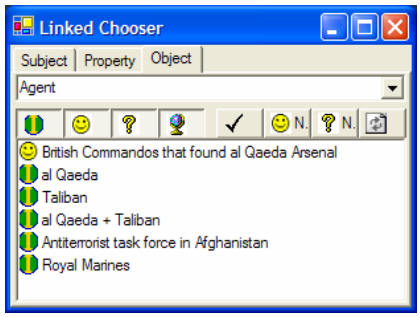
SemanticWord integrates an information extraction system (IES). Automatic information extraction technology promises to significantly reduce the human overhead involved in the semantic annotation task. Although this technology has not reached a level of sophistication required to capture deep relationships in text ([5][7]), it can provide useful annotation fragments. The approach taken in SemanticWord is to supply the tools that would allow users to augment the annotation provided by an IES.

SemanticWord uses AeroDAML, an IES developed at Lockheed Martin [7]. AeroDAML processes text and produces DAML markup that relates instances and values to Ontology classes and types. AeroDAML relies on a high performance commercial information extraction system called AeroText. The default AeroDAML is based in the default AeroText which includes "domain independent" extraction rules capable of extracting many proper nouns and frequently occurring relations. AeroText and consequently AeroDAML can be tailored to particular domains through training sessions with annotated corpuses.

SemanticWord provides an environment for refining and augment the result of IESs. We observed that the default AeroDAML does a good job at recognizing and categoriz-



Property Chooser



Instance Chooser (Object)

Figure 3. Property and Instance Choosers.

The choices correspond to the filling of the property and object columns of the second row of the triples table of Figure 2. The listed choices are constrained by the content of the other cells of the selected triple. These filters can be relaxed by toggling the buttons on the top toolbars. The Instance chooser also supports the definition of new instances.

ing proper nouns but their classification tends to be overly general. It also fails to recognize most of the relations between instances. For example, AeroDAML succeeds in classifying **Kabul** as a **Place** but failed in finding the more specific class **City**, perhaps because there was nothing in the text that might clue AeroDAML about this fact. SemanticWord let AeroDAML to recognize and classify proper nouns but expects the user to refine the classification and to specify their relationships.

SemanticWord drives the information extraction process on the fly. As the user types the content of the document, a background thread feeds new or modified text to AeroDAML in paragraph units (roughly), obtains the extracted entities with their position in the text, and underlines those text regions with a blue wiggly line. This procedure is carried out in a way that resembles Word spelling and grammar checking and is implemented in terms of Microsoft SmartTags technology.

The user can examine the extracted entities and convert them into instance reference annotations. As part of this conversion the user has the option of refining the extracted type. Once an extracted entity has been transformed into an instance reference it behaves just like a natively created instance reference. In particular, it can be dragged and dropped onto cells of triple bags to describe the relationships that AreoDAML missed.

Annotated Templates

Annotated text templates reduce the amount of work involved in authoring both semantic annotations and document content. A template consists of a text fragment annotated with semantic and template related descriptions, and persists as a (typically quite small) word document.

A template may be inserted into a document just like any other document. Both the text and annotations of the template are copied into the target document. After insertion, the copy can still be subjected to further editing and annotating.

Templates are authored in SemanticWord in template design mode. All annotations tools described previously are also available for annotating templates in template design mode plus an additional toolbar that includes the template specific authoring tools described below. We expect that non-programmers would be able to author templates.

Instance Placeholder: An instance placeholder annotates a region of text that needs be replaced by an instance reference when the template is used in a document. It also serves as the surrogate for an instance reference, and as such, it can participate as the subject or the object of one or more triples in the template's triple bags.

An instance placeholder is rendered like an instance reference annotation but with a different icon. In design mode this icon can be dragged over triple tables to compose the semantic annotations that describe the template. It can also be dragged over another instance placeholder to specify a co-reference requirement. In instantiation mode, this icon is a drop site for the concrete instance that is going to be bound to the instance placeholder.

When an instance placeholder is bound to an instance reference, the label of the instance reference replaces the template's text and all co-referential instance placeholders are bound to that instance.

Optional group: An optional group delimits a region of

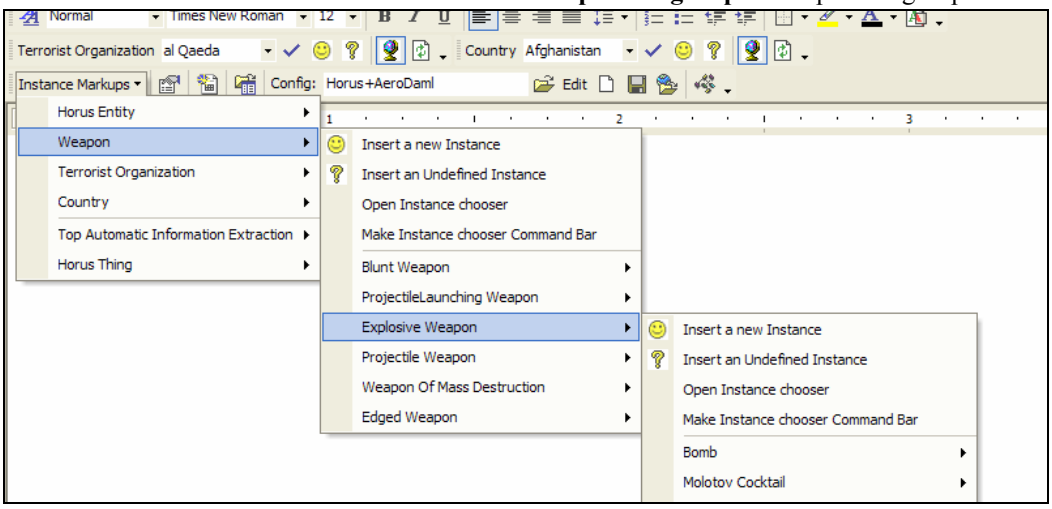


Figure 4. Toolbars and Menus.

The last two toolbar rows belong to SemanticWord. The first row contains two juxtaposed *personal class toolbars*. The first one is tied to the class "Terrorist Organization" and has selected the instance "al Qaeda". The second one is tied to "Country" and has selected "Afghanistan". Clicking in the Check button will generate both the text and the annotation corresponding to the selected instance. The other buttons are for defining new instances before inserting their text and annotation. The last toolbar row has its *classes cascading menu* opened. This menu provides access to several class related functions. The most recently chosen classes get added to the top of the menu (like Weapon, Terrorist Organization, and Country) for easy access.

text in the template that can be optionally included in the instantiation of the template. The text delimited by an optional group can contain annotations and other groups. In particular, it can contain instance placeholders. Opting to delete an optional group from an instantiated template will automatically remove any triples having a cell linked to an instance placeholders within the deleted group.

Repeated group: Like an optional group, repeated group annotation delimits a region of text and can also contain other groups and annotations. During instantiation the user can ask that a repeated group be replicated any number of times. Each replication of the group creates its own incarnation of the instance placeholders that it contains. When the group is replicated, all triples with cells linked to the instance placeholders contained in the group are replicated as well.

The utility of annotated templates is enhanced by the IES described above. The IES analyses the document and generates instance reference annotations corresponding to the concrete entities mentioned in the text. These instance references can be dragged over the template instance place holders to instantiate the template and generate instantiated triples describing their relationships.

ANNOTATING TEXT REGIONS

In SemanticWord, semantic descriptions are distributed throughout the document and attached to text regions that "support" their content. This is not a requirement for the semantic web. Most of the semantic markup authoring tools reported in the literature do not adopt this practice. The descriptions they produce are associated only with a document, not with portions of that document.

We speculate that relating a semantic description to the text that supports it has advantages in terms of annotation authoring, reuse, maintenance, and validation. However, we also recognize that this practice might introduce unnecessary complications.

Some of the advantages of attaching semantic descriptions to text are:

- Descriptions can be reused if the text is reused. Annotations are carried over along text cut/copy and paste operations and when document fragments are reused in other documents.
- Conformity between the semantic descriptions and the content of the document can more easily be validated and maintained.
- Authors might find it natural to find annotations by finding, through familiar text search/scroll mechanisms, the text to which the annotations are attached. Contrast this with browsing the semantic markup directly. For example, in SemanticWord authors compose triples by dragging around instance references placed within the text.
- Markup that is tied to text fragments disappears if the text fragment, or a region containing it, is deleted. Generally, this is desirable because the document's content no longer supports the statement formalized by the deleted annotation.

Among the difficulties of this approach we found:

- If an entity (e.g., a person or place) is mentioned several times within the text, it might be necessary to duplicate its annotation too.
- Some concepts might be implicit or too abstract to be located in the text.
- As changes are made to text within an annotated region – particularly at its boundaries – heuristics must be used to adjust the boundaries. The use of paired brackets for rendering these regions keeps the user informed of the result of these heuristics.

Although SemanticWord is biased toward the attachment semantic annotations to text, it does not mandate it, opening a whole spectrum of hybrid compromises. For example, authors might choose to attach instance references to text but to describe their relationships in a single global triple bag. Moreover, not even the instance reference annotations are required because the triples can be filled directly from instance choosers. More serious use of SemanticWord will be required to weigh the pros and cons of this approach..

RELATED WORK

Research in semantic annotations is still in its infancy. A number of systems have been developed to date that demonstrate different capabilities. However, the approaches adopted by these systems do not necessarily compete against each other but rather address different issues.

Ont-O-Mat ([4][5]), one of the first annotation systems to appear, is the concrete implementation of CREAM [4], an annotation and content authoring framework conceived for the easy creation of relational metadata (i.e., relations between instances). Ont-O-Mat includes its own HTML *document editor* for viewing and composing the content of the document being annotated and an *ontology and fact browser* for visualizing the markup collected by a crawler and for authoring the markup that annotates a document. Like SemanticWord, Ont-O-Mat also provides mechanisms that simplify the creation of markup, document content, or both. For example, dragging text from the document editor and dropping it on top of a class listed in the ontology and fact browser could automatically create an instance of that class with the dragged text filling some property of the created instance (e.g., its name). Similarly, dragging an instance listed in the ontology and fact browser and dropping it at some location within the document editor could insert in that location the text corresponding to the filler of some property of the dropped instance and eventually could attach to that text a hyperlink that describes the instance further. A *meta ontology* specifies the type of ac-

tions to be carried out through the dragging and dropping operations.

S-CREAM [5] extends the CREAM framework with an information extraction component for the semi-automatic generation of annotations. In S-CREAM manual annotation is supported by Ont-O-Mat while automatic information extraction is supported by Amilcare [1], an adaptive information extraction system (IES). Because the IES is unable to capture relationships in a graph that connects the individuals described in the text, the output of the IES has to be mapped into a Discourse Representation (dependent on the domain) before generating a set of markup hypotheses. This technique is still very rudimentary.

SMORE [6] provides an environment for composing the content and the inline semantic annotation of web pages, email, and other online documents. Like SemanticWord, SMORE aims to support semantic annotation without disrupting the document creation process. Toward this end SMORE supports practices like using place holders to defer the final determination of the markup, referencing multiple ontologies that can be brought to bear when the need arises, and extending ontologies if none of the known ontologies fit the user needs. SMORE also integrates several unique capabilities, like the ability to annotate parts of images using SVG, an advance ontology search capability, web scraping, and a Semantic Virtual Portal that provides links to semantically related material

MnM [8] and Melita [2] are environments that streamline the automatic production of semantic annotations using an information extraction system (IES). The process supported by these systems comprises several activities, including manually annotating web pages (for training the IES), training the IES using the annotated pages, tuning the performance of the trained system, and running the IES to automatically annotate a set of pages. MnM implements a generic process model which is also generic with respect to the specific ontology server and information extraction tool used. Melita is a demonstration system that seamlessly integrates manual annotation, incremental training, and automatic information extraction in a timely and non-intrusive way. These systems have demonstrated that is possible to highly automate the generation of semantic annotations. Unfortunately, the scope of these annotations is restricted to only filling in one information template per document. Both systems use Amilcare as their IES.

Among the described annotation tools only SemanticWord provides an environment for document authoring and semantic annotation that extends a COTS product that authors have already adopted (MS Word). SemanticWord is also the only one that associates semantic annotations within text regions and consequently facilitates annotation reuse and maintainability.

CONCLUSIONS

SemanticWord integrates into a widely used COTS product an environment for authoring document content and annotations. It includes several features intended to minimize the cost involved in authoring semantic annotations: customizable tools for generating content and annotations simultaneously, direct manipulation of annotations embedded in the document, reusable annotations, annotated text templates, and an information extraction system including support for refining and augmenting its output.

REFERENCES

[1] Ciravegna, F. *Adaptive Information Extraction from Text by Rule Induction and Generalisation*, Proc. of 17th International Joint Conference on Artificial Intelligence (IJCAI 2001) , Seattle, August 2001.

[2] Ciravegna, F., Dingli, A., Petrelli, D., and Wilks, Y., *Timely and Non-Intrusive Active Document Annotation via Adaptive Information Extraction*, in Semantic Authoring, Annotation & Knowledge Markup (SAAKM 2002), ECAI 2002 Workshop, July 22-26, 2002 , Lyon, France.

[3] Connolly, D, van Harmelen, F., Horrocks, I, McGuinness, D., Patel-Schneider, P., and Stein, L. *DAML+OIL (March 2001) ReferenceDescription*, W3C Note 18 December 2001 http://www.w3.org/TR/daml+oil-reference.

[4] Handschuh, S and Staab, A. *Authoring and Annotation of Web Pages in CREAM,* in Proceedings of the WWW2002 - Eleventh International World Wide Web Conference, Hawaii, USA, May 2002.

[5] Siegfried Handschuh, Stephen Staab, Fabio Ciravegna, *S-CREAM -- Semi-automatic CREAtion of Metadata*, in Semantic Authoring, Annotation & Knowledge Markup (SAAKM 2002), ECAI 2002 Workshop, July 22-26, 2002 , Lyon, France.

[6] Aditya Kalyanpur, James Hendler, Bijan Parsia, Jennifer Golbeck, *SMORE - Semantic Markup, Ontology, and RDF Editor*, available at http://www.mindswap.org/papers/SMORE.pdf

[7] Paul Kogut, William Holmes, *AeroDAML: Applying Information Extraction to Generate DAML Annotations from Web Pages* , First International Conference on Knowledge Capture (K-CAP 2001), Workshop on Knowledge Markup and Semantic Annotation, Victoria, B.C. October 21, 2001 AeroDAML.

[8] Maria Vargas-Vera, Enrico Motta, John Domingue, Mattia Lanzoni, Arthur Stutt1, and Fabio Ciravegna, *MnM: Ontology-Driven Tool for Semantic Markup,* in Proceedings of ECAI 2002, July, 2002 , Lyon, France.